3D Neural Edge Reconstruction

Lei Li[1] Songyou Peng[1,2†] Zehao Yu[3,4] Shaohui Liu[1] Rémi Pautrat[1,6]
Xiaochuan Yin[5] Marc Pollefeys[1,6]
[1]ETH Zurich [2]MPI for Intelligent Systems, Tübingen [3]University of Tübingen
[4]Tübingen AI Center [5]Utopilot [6]Microsoft
neural-edge-map.github.io

Abstract

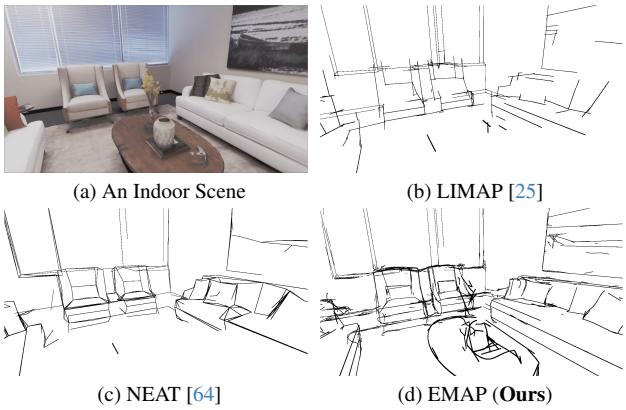
Real-world objects and environments are predominantly composed of edge features, including straight lines and curves. Such edges are crucial elements for various applications, such as CAD modeling, surface meshing, lane mapping, etc. However, existing traditional methods only prioritize lines over curves for simplicity in geometric modeling. To this end, we introduce EMAP, a new method for learning 3D edge representations with a focus on both lines and curves. Our method implicitly encodes 3D edge distance and direction in Unsigned Distance Functions (UDF) from multi-view edge maps. On top of this neural representation, we propose an edge extraction algorithm that robustly abstracts parametric 3D edges from the inferred edge points and their directions. Comprehensive evaluations demonstrate that our method achieves better 3D edge reconstruction on multiple challenging datasets. We further show that our learned UDF field enhances neural surface reconstruction by capturing more details.

1. Introduction

The straight line belongs to men, the curved one to God.
— Antonio Gaudí

This sentiment is evident in the visual composition of our environments. While straight lines are common in man-made scenes such as walls, windows, and doors [25], curves are more general and ubiquitous from cups, bridges, architectures, to Gothic arts. *Edges*, which are composed of both lines and curves, are the fundamental elements of visual perception. Therefore, accurate edge modeling is crucial for understanding the geometry and structure of our 3D world.

Conventional approaches on 3D reconstruction typically involve inferring dense geometry and abstracting meshes from 2D images [15, 34, 48, 50, 67, 69]. However, the presence of 3D edges offers substantial advantages. First, edges are naturally compact representations that capture the salient features oftentimes around geometric boundaries, which are good indicators for more lightweight and adaptive meshing and 3D modeling with comparably less redundancy. Secondly, in contrast to dense surface modeling from images, 3D edges are unaffected to illumination changes, thus exhibiting better reliability on multi-view reconstruction. Last but not least, 3D edges serve as a universal representation in real-world scenarios, and can be potentially integrated into many applications such as lane mapping [10, 22, 42, 43, 56], motion forecasting [12, 13, 51], medical imaging [45], etc.



(a) An Indoor Scene (b) LIMAP [25] (c) NEAT [64] (d) EMAP (**Ours**)

Figure 1. **Example 3D edge reconstruction on Replica [53].** While prior methods such as LIMAP [25] and NEAT [64] only reconstruct distinctive line segments, our method generates a more complete 3D edge map combining both line and curve features.

The reconstruction of 3D edges is conventionally approached by matching their 2D observations across views. While Bignoli et al. [5] proposed edge point matching using the sparse map from Structure-from-Motion (SfM), it is inherently ill-posed due to its heavy reliance on cross-view edge correspondences, which are generally sparse and prone to ambiguity. Recent works have also improved the quality of 3D line reconstruction [18, 25, 61, 64], but primarily excel in specific scenes where straight lines domi-

† Corresponding author

nate. While general real-world environments with curved structures pose more challenges, recent progress on 2D detection and matching is mostly limited to point and line features and thus inapplicable to such scenarios.

A recent work NEF [70] made a significant step forward in learning 3D curves from multi-view 2D edge observations. Inspired by the recent success of neural radiance field (NeRF) [32], they introduce a neural edge density field and show decent results in reconstructing edges for simple objects. Nevertheless, their proposed edge density field has an inherent bias in edge rendering, leading to less accurate reconstruction. Moreover, its fitting-based edge parameterization process not only requires tedious tuning to specific data, but also struggles with its scalability to larger and more complex scenes. This motivates us to develop a more robust system for 3D edge mapping from 2D observations, which would benefit a wide range of downstream tasks.

Towards this goal, we introduce *EMAP*, a novel approach for accurate 3D edge reconstruction from only 2D edge maps. EMAP comprises the following steps. Firstly, we learn the neural unsigned distance function (UDF) to implicitly model 3D edges, utilizing an unbiased rendering equation to mitigate the inaccuracies observed in NEF. Secondly, once learned, we can obtain the unsigned distance and normal for each point in the space, so a set of precise edge points with directions can be extracted. Finally, based on the guidance of every edge point's location and direction, we design a simple yet robust algorithm for parametric line and curve extraction, that can be applied across various challenging scenarios. Our comprehensive evaluations of EMAP, from synthetic CAD models to real-world indoor and outdoor scenes, show its superior performance in 3D edge reconstruction. In addition, we also observe that initializing the optimization process of the recent neural implicit surface reconstruction method with our trained UDF field enables the reconstructing of better details.

Overall, the contributions of this paper are as follows:

- We propose EMAP, a 3D neural edge reconstruction pipeline that can learn accurate 3D edge locations and directions implicitly from multi-view edge maps.
- We develop a 3D edge extraction algorithm to robustly connect edge points with edge direction guidance.
- We show that our model can generate complete 3D edge maps and help optimize dense surfaces.

2. Related Work

Geometry-based 3D Line Reconstruction. As a pioneering work, Bartoli and Sturm [4] introduces a full SfM system using line segments, which is later improved under manhattan assumption [47] and in stereo systems [8]. Recently, with the developments of line detections [36, 37, 65] and matching [2, 36, 38] thanks to the advent to deep learning, several works have attempted to revisit the line mapping problem through graph clustering [19], leveraging planar information [61] and incorporating into SLAM systems [17, 23, 29, 52, 77]. In particular, recent work LIMAP [25] introduces a robust 3D line mapping system with structural priors which can adapt to different existing line detectors and matchers. Despite these advances, all the works are limited to straight lines and often produce segmented small lines when it comes to curves. In contrast, edges are generally easier to detect and are redundantly present in most scenes. In this project, rather than relying on lines, we build our 3D mapping system using robust 2D edge maps.

Learning-based 3D Line/Curve Reconstruction. In contrast to geometry-based methods, some approaches [26, 59, 74] shifted their focus to directly extract parametric curves from given edge point clouds. Typically, they require keypoint detection, clustering, and linkage. Even under the relaxed setting, it is still challenging to generate clean parametric curves due to the complex connectivity of curves and imperfect point clouds [70]. To address this limitation, NEF [70] integrates NeRF [32] for edge mapping from multi-view images, extracting 3D curves from the learned neural edge field through a carefully designed post-processing. While NEF achieves decent performance on CAD models, it is constrained to simple and low-precision object-level edge mapping. A concurrent work, NEAT [64], utilizes VolSDF [69] to build dense surfaces and incorporates a global junction perceiving module to optimize 3D line junctions with 2D wireframe supervision. Although NEAT can produce 3D wireframes, it is restricted to modeling line segments only. Additionally, their need for textured objects is a limitation. By contrast, we use the unisgned distance function (UDF) to represent edges, enabling the construction of both line segments and curves without the necessity for target textures. We further show that our method can faithfully reconstruct edges for complex scenes.

Neural Implicit Representations. Neural implicit representations have emerged as a powerful tool for a spectrum of computer vision tasks, including object geometry representation [9, 24, 30, 34, 35, 40, 46, 57, 62, 66, 68], scene reconstruction [6, 20, 39, 71, 72, 75, 76], novel view synthesis [28, 32, 44, 73] and generative modelling [7, 33, 49]. Recent works [27, 58, 60, 69, 72] show impressive high-fidelity reconstruction by learning the implicit signed distance function (SDF). However, the SDF representation constrains to modeling closed, watertight surfaces. In contrast, NeuralUDF [3] exploits UDF to represent surfaces, offering a higher degree of freedom to represent both closed and open surfaces. We find UDF as a suitable representation to model edges implicitly, in comparison to SDF used in NEAT [64] and edge volume density from NEF [70].

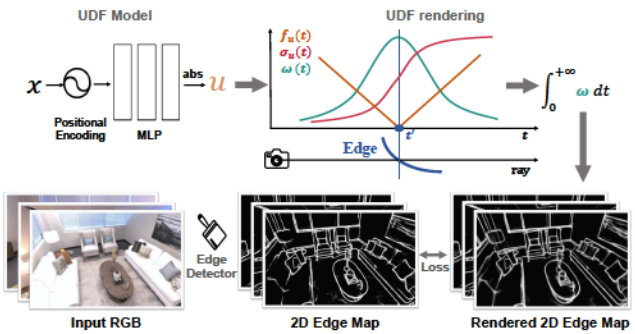


Figure 2. **UDF learning overview.** We utilize a vanilla NeRF [32] MLP that outputs absolute values to model the 3D UDF field. Edge maps are rendered using a density-based edge neural rendering technique, combined with an unbiased UDF rendering approach to eliminate bias. Our primary supervision comes from 2D edge maps predicted by a pre-trained edge detector.

3. Method

Our goal is to build a 3D edge map from multi-view posed 2D edge maps. To this end, we first introduce our edge representation and edge field learning in Sec. 3.1. Next, we present our 3D parametric edge extraction from the learned edge representations in Sec. 3.2.

3.1. Edge Field with Unsigned Distance Functions

Multi-view Edge Maps. Since edge maps are generally invariant to illumination changes and are more robustly detected across various scenes than lines, our method utilizes multiple posed 2D edge maps as inputs. We apply pre-trained edge detectors to predict an edge map E for each input RGB image. Each pixel of E has a value within $[0, 1]$, indicating its probability of being an edge.

Density-based Edge Neural Rendering. We use an unsigned distance function (UDF) to represent edges, denoted as f_u. This function computes the unsigned distance from a given 3D point to the nearest edge. The UDF is defined as:

$$f_u : \mathbb{R}^3 \to \mathbb{R} \quad \mathbf{x} \mapsto u = \mathrm{UDF}(\mathbf{x}) , \tag{1}$$

where x is a 3D point and u is the corresponding UDF value.

To render an edge pixel in a certain view, we trace a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$. This ray originates from the camera's center $\mathbf{o}$ and extends in direction $\mathbf{d}$ [32]. To apply volume rendering for edge modeling, it is necessary to establish a mapping Ω_u [27, 58] that transforms the distance function $f_u(\mathbf{r}(t))$ into volume density $\sigma_u(t)$ as

$$\sigma_u(t) = \Omega_u(f_u(\mathbf{r}(t))) . \tag{2}$$

In the rendering equation, the transmittance $T(t)$ and weight $\omega(t)$ along the camera ray $\mathbf{r}$ are accumulated as

$$T(t) = \exp\left(-\int_0^t \sigma_u(v)dv\right), \quad w(t) = T(t) \cdot \sigma_u(t) . \tag{3}$$

To effectively handle appearance changes under different viewing angles, most neural field-based surface reconstruction [34, 58, 67, 71] disentangles geometry and appearance. In contrast, edge maps are generally unaffected by lighting, making them view-independent. Therefore, this simplifies the rendering process for edge maps. as it only requires the accumulation of view-independent, density-based weights w along a ray $\mathbf{r}$. Now, the rendered edge value $\hat{E}$ along ray $\mathbf{r}$ is formulated as:

$$\hat{E}(\mathbf{r}) = \int_0^{+\infty} w(t)dt = 1 - T(+\infty) , \tag{4}$$

Eq. (4) establishes the connection between rendered edge values and the transmittance at the end of the camera rays. Intuitively, this means that the rendered edge value is 1 when the camera ray hits an edge in 3D space, and 0 otherwise. Please refer to the supplements for more details.

Unbiased Density Functions for UDF Rendering. NEF [70] also uses volume rendering for rendering edges. Unlike ours, they utilize edge density to represent edges and an additional network to predict edge values. However, this approach introduces an inherent bias in edge rendering. Similar to the naive solution presented in NeuS [58], the issue comes from the weight function w in Eq. (3), where its local maximum does not coincide with the actual intersection point of the camera ray and the edges.

To address this issue, we incorporate unbiased UDF rendering [27] into our density-based edge rendering framework. As proved in NeuS, density function σ_u should increase monotonically to make the weight function unbiased. However, UDF values are not monotonous along a ray [27]. To adapt the unbiased density function Ω_s, which is originally induced in NeuS [58], for UDF use, the monotonically increased density function σ_u [27] is formulated as

$$\sigma_u(t) = \Psi(t) \cdot \Omega_s\left(f_u(\mathbf{r}(t))\right) + (1 - \Psi(t)) \cdot \Omega_s\left(-f_u(\mathbf{r}(t))\right) , \tag{5}$$

where $\Psi(t)$ is a differentiable visibility function designed in [27] to capture the monotonicity change in UDF. Ψ is 0 behind the intersection point between the camera ray and the hit edge, and is 1 before the intersection point. Besides, $\Psi(t)$ is differentiable around the intersection point to make the UDF optimization more stable.

Ray Sampling Strategy. A key characteristic of 2D edge maps is their significant sparsity, with edges occupying a much smaller area compared to non-edge regions. To enhance training efficiency and stability, we apply an importance sampling strategy for camera rays, with 50% of rays uniformly sampled from edge areas in the edge maps and the remaining 50% from non-edge areas. Such a sampling strategy ensures that our UDF field training is concentrated on edge areas, thereby substantially speeding up the training process. Additionally, our sampling strategy offers an

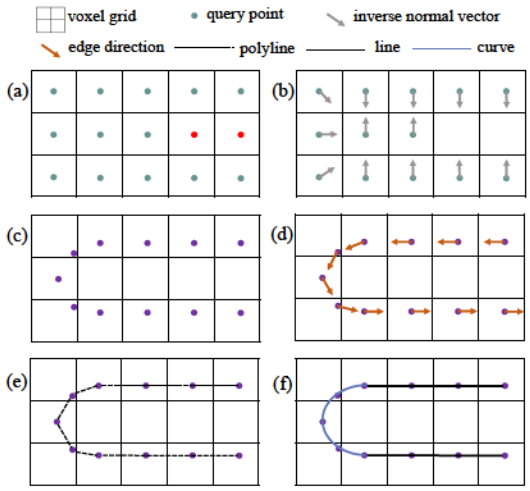


Figure 3. **Illustration of our 3D parametric edge extraction steps.** For simplify, our schematic is depicted in the 2D plane. Our 3D edge extraction algorithm comprises five main stages: point initialization (a), point shifting (b to c), edge direction extraction (c to d), point connection (d to e), and edge fitting (e to f).

elegant solution to the issue of occlusion, a challenge noted in [70]. The rendered edge maps might contain edges not present in the input edge images due to occlusion. In contrast to the complicated occlusion handling strategy introduced in [70], our approach inherently alleviates this challenge by focusing the training on points from the visible edges presented in the input edge maps.

Loss Functions. The total loss function can be written as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{edge}} + \lambda \mathcal{L}_{\text{eik}} \,, \tag{6}$$

where $\mathcal{L}_{\text{edge}}$ represents the Mean Square Error (MSE) between the rendered and input edge images. $\mathcal{L}_{\text{eik}}$ denotes the Eikonal loss [16], which promotes the learned UDF to be physical distance. λ is used to balance these losses.

3.2. 3D Parametric Edge Extraction

With UDF learning, edge locations are implicitly encoded within the UDF field. However, accurately extracting edge points from the UDF field is non-trivial due to the absence of a real zero-level set in the UDF field. Additionally, formulating these edge points into parametric edges poses significant challenges due to the complex connections of edges. To extract points from the learned density field, NEF [70] selects points with edge density values greater than a specified threshold, ϵ. This approach leads to an approximated edge point set that is ϵ-bounded [27]. While this method effectively generates comprehensive point clouds, the ϵ-bounded point set does not align accurately with the actual edge locations.

To eliminate the error in edge point extraction, we leverage the physical property of UDF that reflects real-world

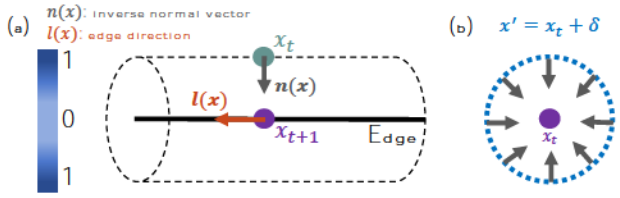


Figure 4. **Illustration of the overview (a) and the cross-section (b) of UDF field.** (a) In UDF field, edge points are ideally located at the zero-level set, with UDF values being larger away from these points. A query point x_t can be precisely shifted to a more accurate position x_{t+1} by following the UDF value and the inverse normal vector $n(x)$. The edge direction $l(x)$ aligns with the tangent direction at the edge point x_{t+1}. (b) The inverse normal vectors of all surrounding points on the cross section are pointing towards the query point.

distances to the edges. Specifically, we develop a 3D edge extraction algorithm composed of five main stages: point initialization, point shifting, edge direction extraction, point connection, and edge fitting, as illustrated in Fig. 3. This algorithm takes the trained UDF field as input and outputs parametric 3D edges, including line segments and curves.

Point Initialization. Under eikonal loss supervision, the optimized UDF values represent physical distances to the nearest edges. To initialize potential edge points, we begin with the center points of all voxel grids and obtain their UDF values from the UDF field. Subsequently, we eliminate query points whose UDF values exceed a specified threshold ϵ' (red points in Fig. 3 (a)).

Point Shifting. As illustrated in Fig. 4 (a), the normalized inverse gradient of the UDF field indicates the inverse normal vector pointing towards edges. Drawing inspiration from OccNet [31], we refine the point x iteratively towards the edge using its distance and inverse normal direction:

$$x_{t+1} \Leftarrow x_t - f_u(x_t) \cdot \frac{\nabla f_u(x_t)}{\|\nabla f_u(x_t)\|} \,, \tag{7}$$

where t denotes the t-th iteration. As a result of this iterative process, the initial points converge to the edge center (from Fig. 3 (b) to Fig. 3 (c)).

Edge Direction. Establishing connections between edge points is a crucial step in constructing parametric edges. While most methods [11, 42, 70] estimate parameters through least-squares fitting of lines/curves on extracted points, this fitting-based approach for edge extraction is not always robust or accurate. In contrast, inspired by [37, 63], we find that combining the edge direction field with the edge distance field can robustly produce edge parameters. Given that inverse normal vectors invariably point towards edges (see Fig. 4 (b)), we first devise an edge direction extraction method based on this set of inverse normal vectors. Specifically, for a query point x, we introduce minor shifts

set $\{\delta\}_N$ with size of N to generate an adjoining point set $\{x'\}_N$, where $\{x'\}_N = x + \{\delta\}_N$. The inverse normal vectors of these points, denoted as $\{n\}_N$, are obtained from the learned UDF field. The edge direction, denoted as l, is identified as the null space of $\{n'_i\}$, since the edge direction is perpendicular to all inverse normal vectors in $\{n\}_N$. Therefore, l can be extracted with singular value decomposition (SVD):

$$A = U\Sigma V^T, \quad l = V[:, \text{argmin}(\Sigma)], \tag{8}$$

where A is the matrix representation of $\{n\}_N$ and l corresponds to the eigenvector associated with the smallest eigenvalue. Note that N should be sufficiently large to ensure the stability of the extracted edge direction. Unlike DeepLSD [37], we can obtain a precise edge direction field without relying on any 2D direction supervision.

Point Connection. After accurately determining the edge point locations and directions, we proceed to connect these edge points guided by the edge direction to create polylines (Fig. 3 (d) to (e)). Specifically, we begin by selecting candidate points and then compute directional errors for points adjacent to these candidates. Based on these directional errors, candidate points are connected to its best-matched neighboring point that growing direction aligns best with its extracted edge direction, i.e., with minimal directional error. This process is repeated, extending the edge polylines progressively until no further growth is possible. To ensure efficiency and accuracy, a non-maximum suppression step is employed to remove any redundant points that may exist between the current candidate and the best-matched point. Please refer to the supplements for more algorithm details.

Edge Fitting. To further parameterize edges, we categorize the polylines into line segments and Bézier curves (Fig. 3 (f)). Initially, we utilize RANSAC [14] to fit lines from the polylines, and select the line segment that encompasses the highest number of inlier points. Following [25], we apply Principal Component Analysis (PCA) to the inlier points, re-estimate the line segment utilizing the principal eigenvector and the mean 3D point, and project all inlier points onto the principal eigenvector to derive the 3D endpoints. This fitting process is repeated for each polyline until the number of inlier points falls below a minimum threshold. For the remaining sub-polylines, we fit each of them with a Bézier curve that is defined by four control points.

To minimize edge redundancy, we further merge line segments and Bézier curves based on two criteria: the shortest distance between candidate edges and the similarity of curvature at their closest points. For line segments, the shortest distance is the minimal point-to-line segment distance, and curvature similarity is their direction's cosine similarity. For Bézier curves, they are the minimal point-to-point distance and the cosine similarity of the tangent vectors at the nearest points, respectively. Candidate edges are merged only if they meet both criteria. This dual-criterion approach ensures that merging happens only when two edges are both similar and close to each other.

To connect edges, all endpoints of line segments and Bézier curves located within a specified distance threshold are merged into shared endpoints. Furthermore, we implement an optimization step [5, 64] to refine the 3D parametric edges by leveraging 2D edge maps, thereby enhancing edge precision. Specifically, we project 3D parametric edges into edge map frames using camera projection matrices and filter out 3D edges that are not visible in over 90% of views.

4. Experiments

4.1. Experiment Setting

Datasets. We consider four diverse datasets: CAD models (ABC-NEF [70]), real-world objects (DTU [1]), high-quality indoor scenes (Replica [53]), and real-world outdoor scenes (Tanks & Temples [21]). ABC-NEF dataset comprises 115 CAD models, each accompanied by 50 observed images and ground truth parametric edges. We select 82 CAD models, excluding those containing inconsistent edge observations (e.g., cylinders or balls). DTU dataset provides dense ground-truth point clouds and we select 6 objects that meet the multi-view constraints among scans processed by [72]. Following [5], we derive edge points by projecting ground-truth dense points onto images and then comparing them with the observations on 2D edge maps to filter out non-edge points. Replica and Tanks & Temples datasets contain larger scenes. Due to the lack of ground-truth edges, we conduct qualitative comparisons among baselines.

Baselines. We compare with three state-of-the-art baselines for 3D line/curve mapping, including two learning-based methods, NEF [70] and NEAT [64], and one geometry-based method, LIMAP [25].

Metrics. Our evaluation involves first sampling points in proportion to the edge's length and subsequently downsampling these points using a voxel grid with a resolution of 256^3. Following the metrics used in [25, 70], we consider Accuracy (Acc), Completeness (Comp) in millimeters, and Recall (R_τ), Precision (P_τ), F-score (F_τ) in percentage with a threshold τ in millimeters. Moreover, we report Edge Direction Consistency (Norm) in percentage to analyze the precision of edge direction extraction.

Implementation Details. For f_u, we utilize 8-layer Multilayer Perceptrons (MLPs). Each layer in the MLP contains 512 neurons for larger scenes, such as Tanks & Temples, and 256 neurons for other datasets. We sample 1024 rays per batch, among these rays, 512 rays are sampled from edge areas. We train our model for $50k$ iterations on ABC-NEF dataset, and $200k$ iterations on other datasets. We train

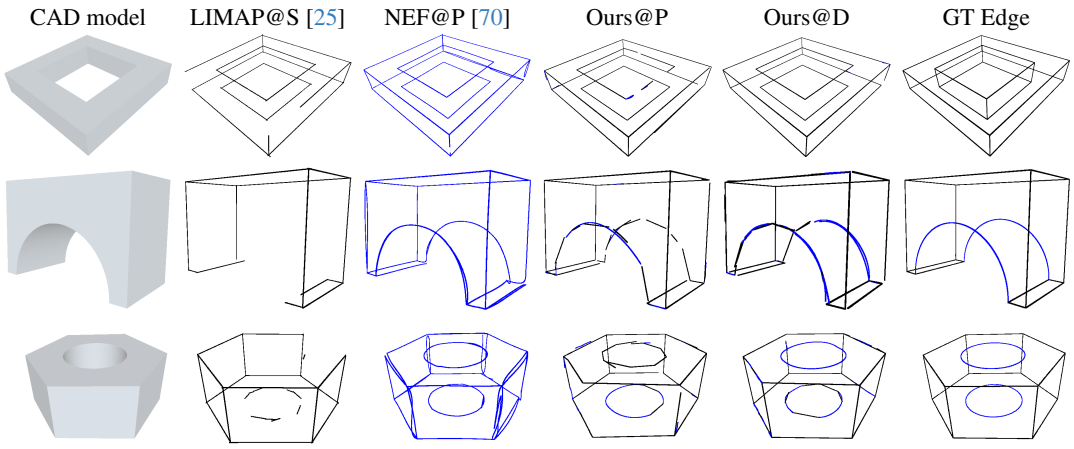


Figure 5. **Qualitative comparisons on ABC-NEF [70].** Lines are shown in black and curves in blue. Thanks to our precise edge extraction capabilities for both lines and curves, we achieve complete and accurate modeling of these elements.

Method	Detector	Modal	Acc↓	Comp↓	Norm↑	R5↑	R10↑	R20↑	P5↑	P10↑	P20↑	F5↑	F10↑	F20↑
LIMAP [25]	LSD	Line	9.9	18.7	94.4	36.2	82.3	87.9	43.0	87.6	93.9	39.0	84.3	90.4
	SOLD2	Line	**5.9**	29.6	90.1	**64.2**	76.6	79.6	**88.1**	**96.4**	**97.9**	**72.9**	84.0	86.7
NEF [70]	PiDiNet†	Curve	11.9	16.9	90.9	11.4	62.0	91.3	15.7	68.5	96.3	13.0	64.6	93.3
	PiDiNet	Curve	15.1	16.5	89.7	11.7	53.3	89.8	13.6	52.2	89.1	12.3	51.8	88.7
	DexiNed	Curve	21.9	15.7	85.9	11.3	48.3	87.7	11.5	39.8	71.7	10.8	42.1	76.8
Ours	PiDiNet	Edge	9.2	15.6	93.7	30.2	75.7	89.8	35.6	79.1	95.4	32.4	77.0	92.2
	DexiNed	Edge	8.8	**8.9**	**95.4**	56.4	**88.9**	**94.8**	62.9	89.9	95.7	59.1	**88.9**	**94.9**

Table 1. **Edge reconstruction results on ABC-NEF [70].** Results from NEF's released pretrained models are indicated by †. Our method surpasses all others in terms of completeness and achieves accuracy comparable to LIMAP [25].

our network with the Adam optimizer with a learning rate of 5×10^{-4}, while the UDF model f_u is trained with a learning rate of 1×10^{-4} and initialized with sphere initialization [66]. For edge detection for NEF and ours, we consider PiDiNet [54] and DexiNed [41]. PiDiNet [54] is employed for indoor scenes, such as DTU and Replica, due to its superior performance in these settings. Conversely, DexiNed [41] is applied to outdoor scenes, as it is primarily trained on outdoor scenes. On the synthetic ABC-NEF dataset, we show results with both detectors. For LIMAP, we follow their paper and we use SOLD2 [36] for indoor scenes and LSD [55] for outdoor scenes. NEAT is trained with 2D wireframes from HAWPV3 [65].

4.2. Evaluation of 3D Edge Reconstruction

Evaluation on ABC-NEF Dataset. We show the quantitative and qualitative comparisons on Table 1 and Fig. 5. Note that NEAT fails on the ABC-NEF dataset because of its heavy dependence on texture input. NEF demonstrates decent performance at $\tau = 20$. However, their performance drops significantly when τ is set to 10 and 5. This is attributed to its bias in edge rendering and its fitting-based post-processing. LIMAP shows remarkable precision across various τ thresholds. Such consistency stems from its non-linear refinement over multi-view 2D supports. Nonetheless, LIMAP's inability to reconstruct curves leads to lower scores in completeness and recall. Our method, when combined with either of the 2D edge detectors, consistently outperforms all baselines. Notably, as shown in Table 1, combined with the DexiNed detector, our method achieves superior results in completeness, edge direction consistency, recall, and F-Score. We also show competitive accuracy and precision when compared to LIMAP.

Method	Detector	Curve			Line		
		Acc↓	Comp↓	Norm↑	Acc↓	Comp↓	Norm↑
LIMAP [25]	LSD	272.6	50.1	84.8	34.6	11.3	95.9
	SOLD2	295.7	82.2	76.8	**20.0**	18.1	92.1
NEF [70]	PiDiNet†	265.0	27.1	77.9	40.4	13.7	92.6
	PiDiNet	263.1	23.9	77.6	43.9	14.0	91.4
	DexiNed	250.5	20.3	72.6	56.2	13.8	87.3
Ours	PiDiNet	253.7	25.7	88.1	43.1	12.8	93.7
	DexiNed	**241.0**	**10.9**	**88.7**	46.7	**7.7**	**95.4**

Table 2. **Accuracy, completeness and normal consistency results with curves and lines on ABC-NEF [70].** Our method with DexiNed edge detector yields overall the strongest performance on curves among all baselines.

To further analyze the performance of different edge types, we classify the ground truth edges into curves (including BSplines, ellipses, and circles) and line segments, based on the GT annotations. We provide accuracy, completeness, and edge direction consistency in Table 2 to analyze the separate reconstruction abilities for curves and lines. Note that these results are computed based on all predictions specific to either curves or lines, as other methods do not differentiate between these two types of edges. We

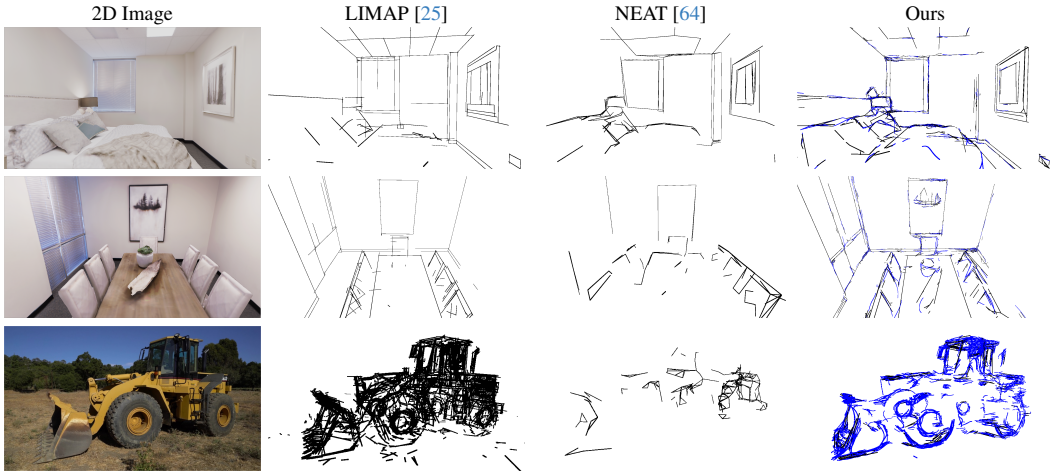


Figure 6. **Qualitative comparisons on the Replica [53] and Tanks & Temples [21] datasets.** The first two scenes are from the Replica dataset, while the last scene is from the Tanks & Temples dataset.

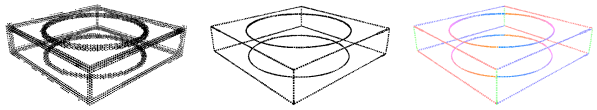
Figure 7. **Qualitative comparisons on DTU [1].** Our results demonstrate complete edge structure, whereas other methods result in redundant line segments or imprecise curves.

Scan	LIMAP [25]		NEF [70]		NEAT [64]		**Ours**	
	R5↑	P5↑	R5↑	P5↑	R5↑	P5↑	R5↑	P5↑
37	**75.8**	74.3	39.5	51.0	63.9	**85.1**	62.7	83.9
83	**75.7**	50.7	32.0	21.8	72.3	52.4	72.3	**61.5**
105	**79.1**	64.9	30.3	32.0	68.9	73.3	78.5	**78.0**
110	79.7	65.3	31.2	40.2	64.3	**79.6**	**90.9**	68.3
118	59.4	62.0	15.3	25.2	59.0	71.1	**75.3**	**78.1**
122	79.9	79.2	15.1	29.1	70.0	82.0	**85.3**	**82.9**
Mean	74.9	66.1	27.2	33.2	66.4	73.9	**77.5**	**75.4**

Table 3. **Edge reconstruction results on DTU [1].**

can see that our method with DexiNed exhibits superior results in reconstructing curves. As for line segments, our performance is marginally lower than the best-performing method LIMAP which is specially optimized for lines.

Evaluation on DTU Dataset. Our assessment of the DTU dataset, as outlined in Table 3 and Fig. 7, shows our proficiency in real-world scenarios. Notably, our approach achieves the highest recall and precision among all baselines. The DTU dataset presents a challenging scenario for edge extraction due to its varying lighting conditions. However, our edge refinement step proves effective in preserving primary edges, a point we elaborate on in Sec. 4.3. Fig. 7 shows LIMAP tends to produce redundant line segments, leading to high recall but reduced precision. NEF's post-processing is sensitive to different scenes, resulting in noisy edge fitting. NEAT, despite producing clean outputs, its inability to handle curves constrains its overall performance.



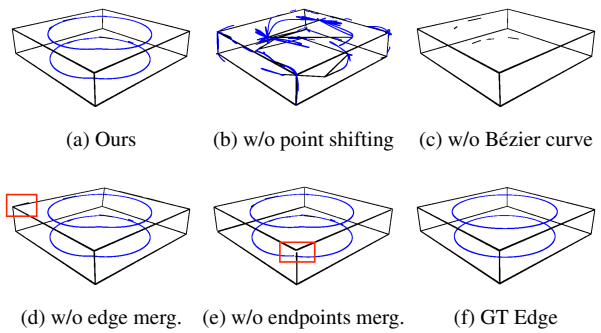
(a) w/o point shifting (b) w/ point shifting (c) edge direction

Figure 8. **Visualization of point shifting and edge direction.** Edge points are shown in point clouds and edge directions in color. The point shifting step significantly refines the locations of edge points. The edge extraction step yields accurate results, as seen in parallel lines sharing the same direction and curves exhibiting continuously changing directions.

Qualitative Evaluation on Indoor & Outdoor Scenes. To really showcase the power of our method in capturing scene-level geometry, we further run our method on indoor and outdoor scenes. Note that since NEF is not able to produce meaningful reconstructions on larger scenes, we only compare with LIMAP and NEAT. As shown in Fig. 1 and Fig. 6, NEAT, due to its reliance on high-quality surface reconstruction, faces limitations in scene reconstruction, while LIMAP and our method both successfully capture good scene geometry.

4.3. Ablations and Analysis

Parametric Edge Extraction. To better understand our parametric edge extraction process described in Sec. 3.2, we visualize our point shifting and edge direction in Fig. 8.



(a) Ours (b) w/o point shifting (c) w/o Bézier curve

(d) w/o edge merg. (e) w/o endpoints merg. (f) GT Edge

Figure 9. **Qualitative ablation on different component of our parametric edge extraction.** The absence of any module in our edge extraction process results in incomplete or noisy qualitative outcomes.

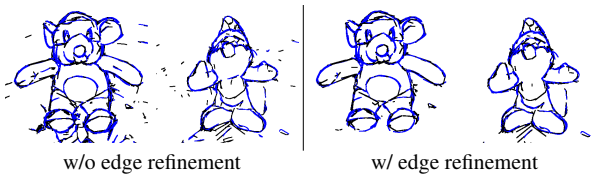
We can clearly see that the extracted point clouds without point shifting are appeared in redundant and inaccurate edge points (Fig. 8 (a)). In contrast, the point shifting step yields point clouds with sharply defined, precise edges (Fig. 8 (b)). In addition, as shown in Fig. 8 (c), the extracted edge directions along parallel lines are consistent, while those on curves vary continuously. This aligns with our expectations.

Furthermore, we also conduct ablation studies in Table 4 and Fig. 9 to evaluate the impact of different components in our edge extraction algorithm. These experiments were performed on the ABC-NEF dataset using the DexiNed detector. First, the removal of the query point shifting step leads to a significant drop in both recall and precision. This indicates that our point-shifting step significantly refines the query points locations. Second, excluding Bézier curves results in a decline in completeness (Fig. 9 (c)), showing that curves are necessary for edge reconstruction. Third, omitting the edge merging step leads to redundant small line segments, as evident in Fig. 9 (d). Finally, the removal of endpoint merging impairs connectivity between edges, as shown in Fig. 9 (e).

	Method	Acc↓	Comp↓	R5↑	P5↑	F5↑
a	Ours	**8.8**	8.9	**56.4**	62.9	**59.1**
b	w/o point shifting	15.3	9.9	29.2	18.7	22.2
c	w/o B'ezier curve	9.4	12.1	54.2	**65.8**	59.0
d	w/o edge merging	10.3	**8.7**	53.8	45.3	48.6
e	w/o endpoints merging	9.3	9.0	51.5	57.7	54.0

Table 4. **Ablation studies on different component of parametric edge extraction on ABC-NEF [70] with DexiNed [41].** Our parametric edge extraction approach with all components achieves the optimal balance between accuracy and completeness.

Edge Refinement. In Fig. 10, we study the effectiveness of our edge refinement module. When input edge maps contain some noises in dark scenes, our initial 3D edge map, without the edge refinement, exhibits some artifacts. However, the edge refinement module markedly mitigates artifacts, achieving a balance between recall and precision.



w/o edge refinement w/ edge refinement

Figure 10. **Ablation study on edge refinement.** Our edge refinement effectively eliminates the majority of noisy edges in background areas.

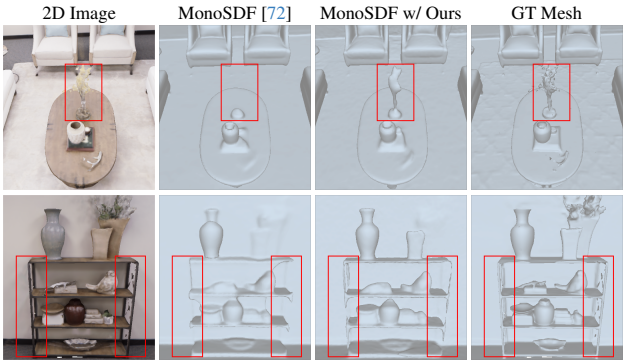


Figure 11. **Dense surface reconstruction on Replica [53].** Utilizing our trained UDF MLP for initialization enables MonoSDF to capture more geometric details, such as the vase in the top row, the shelf in the bottom row.

4.4. Application on Dense Surface Reconstruction

Our method has demonstrated its proficiency in reconstructing 3D edges across a diverse range of scenarios. Building on this success, we further explore the potential of our learned representation to benefit other tasks. A particularly relevant area is dense surface reconstruction.

As shown in Fig. 11, the recent neural-implicit surface reconstruction approach MonoSDF [72] can show decent reconstruction results from only posed multi-view images. However, we notice that they still struggle to capture detailed geometry. To address this, we integrate our method into the MonoSDF pipeline. Specifically, we initialize the geometry MLPs of MonoSDF with our pre-trained UDF MLPs. We can clearly see that such a simple integration can enhance the recovery of geometric details.

5. Conclusions

We introduced EMAP, a 3D neural edge reconstruction pipeline that learns accurate 3D edge point locations and directions implicitly from multi-view edge maps through UDF and abstracts 3D parametric edges from the learned UDF field. Through extensive evaluations, EMAP demonstrates remarkable capabilities in CAD modeling and in capturing detailed geometry of objects and scenes. Furthermore, we show that our learned UDF field enriches the geometric details for neural surface reconstruction.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision (IJCV)*, 2016. 5, 7

[2] Hichem Abdellali, Robert Frohlich, Viktor Vilagos, and Zoltan Kato. L2d2: Learnable line detector and descriptor. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2021. 2

[3] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[4] Adrien Bartoli and Peter Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding (CVIU)*, 2005. 2

[5] Andrea Bignoli, Andrea Romanoni, and Matteo Matteucci. Multi-view stereo 3d edge reconstruction. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. 1, 5

[6] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2

[7] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[8] Manmohan Chandraker, Jongwoo Lim, and David Kriegman. Moving in stereo: Efficient structure and motion using lines. In *Proc. of the International Conf. on Computer Vision (ICCV)*, pages 1741–1748. IEEE, 2009. 2

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[10] Wentao Cheng, Sheng Yang, Maomin Zhou, Ziyuan Liu, Yiming Chen, and Mingyang Li. Road mapping and localization using sparse semantic visual features. *IEEE Robotics and Automation Letters*, 6(4):8118–8125, 2021. 1

[11] Kseniya Cherenkova, Elona Dupont, Anis Kacem, Ilya Arzhannikov, Gleb Gusev, and Djamila Aouada. Sepicnet: Sharp edges recovery by parametric inference of curves in 3d shapes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4

[12] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021. 1

[13] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6797–6806, 2020. 1

[14] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 5

[15] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2007. 1

[16] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020. 4

[17] Yijia He, Ji Zhao, Yue Guo, Wenhao He, and Kui Yuan. Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features. *Sensors*, 18(4):1159, 2018. 2

[18] Manuel Hofer, Michael Maurer, and Horst Bischof. Improving sparse 3d models for man-made environments using line-based 3d reconstruction. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2014. 1

[19] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[20] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[21] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. on Graphics (ToG)*, 2017. 5, 7

[22] Tianyu Li, Li Chen, Xiangwei Geng, Huijie Wang, Yang Li, Zhenbo Liu, Shengyin Jiang, Yuting Wang, Hang Xu, Chunjing Xu, et al. Topology reasoning for driving scenes. *arXiv preprint arXiv:2304.05277*, 2023. 1

[23] Hyunjun Lim, Jinwoo Jeon, and Hyun Myung. Uv-slam: Unconstrained line-based slam using vanishing points for structural mapping. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):1518–1525, 2022. 2

[24] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[25] Shaohui Liu, Yifan Yu, Rémi Pautrat, Marc Pollefeys, and Viktor Larsson. 3d line mapping revisited. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 5, 6, 7

[26] Yujia Liu, Stefano D'Aronco, Konrad Schindler, and Jan Dirk Wegner. Pc2wf: 3d wireframe reconstruction from raw point clouds. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021. 2

[27] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for

multi-view reconstruction of surfaces with arbitrary topologies. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 4

[28] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[29] Daniele Marzorati, Matteo Matteucci, Davide Migliore, and Domenico G Sorrenti. Integration of 3d lines and points in 6dof visual slam by uncertain projective geometry. In *EMCR*. Citeseer, 2007. 2

[30] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[31] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2, 3

[33] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[34] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021. 1, 2, 3

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[36] Rémi Pautrat, Juan-Ting Lin, Viktor Larsson, Martin R Oswald, and Marc Pollefeys. Sold2: Self-supervised occlusion-aware line description and detection. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 6

[37] Rémi Pautrat, Daniel Barath, Viktor Larsson, Martin R Oswald, and Marc Pollefeys. Deeplsd: Line segment detection and refinement with deep image gradients. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 4, 5

[38] Rémi* Pautrat, Iago* Suárez, Yifan Yu, Marc Pollefeys, and Viktor Larsson. GlueStick: Robust image matching by sticking points and lines together. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2023. 2

[39] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2

[40] Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[41] Xavier Soria Poma, Edgar Riba, and Angel Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. 6, 8

[42] Zhijian Qiao, Zehuan Yu, Huan Yin, and Shaojie Shen. Online monocular lane mapping using catmull-rom spline. *arXiv preprint arXiv:2307.11653*, 2023. 1, 4

[43] Tong Qin, Yuxin Zheng, Tongqing Chen, Yilun Chen, and Qing Su. A light-weight semantic map for visual localization towards autonomous driving. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11248–11254. IEEE, 2021. 1

[44] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021. 2

[45] Chris Rorden, Roger Newman-Norlund, Chris Drake, Daniel R Glen, Julius Fridriksson, Taylor Hanayik, and Paul A Taylor. Improving 3d edge detection for visual inspection of mri coregistration and alignment. *bioRxiv*, pages 2022–09, 2022. 1

[46] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019. 2

[47] Grant Schindler, Panchapagesan Krishnamurthy, and Frank Dellaert. Line-based structure from motion for urban environments. In *International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*, 2006. 2

[48] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 1

[49] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[50] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006. 1

[51] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 2022. 1

[52] Fangwen Shu, Jiaxuan Wang, Alain Pagani, and Didier Stricker. Structure plp-slam: Efficient sparse mapping and localization using point, line and plane for monocular, rgb-d and stereo cameras. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*. 2

[53] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl

Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 1, 5, 7, 8

[54] Zhuo Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikäinen, and Li Liu. Pixel difference networks for efficient edge detection. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021. 6

[55] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2008. 6

[56] Huijie Wang, Zhenbo Liu, Yang Li, Tianyu Li, Li Chen, Chonghao Sima, Yuting Wang, Shengyin Jiang, Feng Wen, Hang Xu, et al. Road genome: A topology reasoning benchmark for scene understanding in autonomous driving. *arXiv preprint arXiv:2304.10440*, 2023. 1

[57] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[58] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3

[59] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[60] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[61] Dong Wei, Yi Wan, Yongjun Zhang, Xinyi Liu, Bin Zhang, and Xiqi Wang. Elsr: Efficient line segment reconstruction with planes and points guidance. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[62] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[63] Nan Xue, Song Bai, Fudong Wang, Gui-Song Xia, Tianfu Wu, and Liangpei Zhang. Learning attraction field representation for robust line segment detection. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[64] Nan Xue, Bin Tan, Yuxi Xiao, Liang Dong, Gui-Song Xia, and Tianfu Wu. Volumetric wireframe parsing from neural attraction fields. *arXiv preprint arXiv:2307.10206*, 2023. 1, 2, 5, 7

[65] Nan Xue, Tianfu Wu, Song Bai, Fu-Dong Wang, Gui-Song Xia, Liangpei Zhang, and Philip HS Torr. Holistically-attracted wireframe parsing: From supervised to self-supervised learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2023. 2, 6

[66] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 6

[67] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 3

[68] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[69] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

[70] Yunfan Ye, Renjiao Yi, Zhirui Gao, Chenyang Zhu, Zhiping Cai, and Kai Xu. Nef: Neural edge fields for 3d parametric curve reconstruction from multi-view images. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 4, 5, 6, 7, 8

[71] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. Sdfstudio: A unified framework for surface reconstruction, 2022. 2, 3

[72] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 5, 8

[73] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[74] Xiangyu Zhu, Dong Du, Weikai Chen, Zhiyou Zhao, Yinyu Nie, and Xiaoguang Han. Nerve: Neural volumetric edges for parametric curve extraction from point cloud. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[75] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[76] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2024. 2

[77] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang. Robust visual slam with point and line features. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2017. 2